

Laboratorio Nro. 2

Notación O grande

Objetivo

Corroborar los resultados teóricos obtenidos con la notación asintótica O

Consideraciones iniciales

Leer la Guía



Antes de comenzar a resolver el presente laboratorio, leer la **“Guía Metodológica para la realización y entrega de laboratorios de Estructura de Datos y Algoritmos”** que les orientará sobre los requisitos de entrega para este y todos los laboratorios, las rúbricas de calificación, el desarrollo de procedimientos, entre otros aspectos importantes.

Registrar Reclamos



En caso de tener **algún comentario** sobre la nota recibida en este u otro laboratorio, pueden **enviarlo** a través de <http://bit.ly/2g4TTKf>, el cual será atendido en la menor brevedad posible.

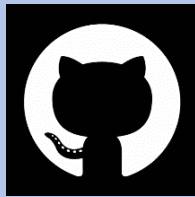
Traducción de Ejercicios



En el GitHub del docente, encontrarán la traducción al español de los enunciados de los Ejercicios en Línea.

**Visualización de
Calificaciones**

A través de **Eafit Interactiva** encontrarán un **enlace** que les permitirá **ver un registro de las calificaciones** que **emite el docente** para cada taller de laboratorio y según las rubricas expuestas. **Véase sección 3, numeral 3.8.**

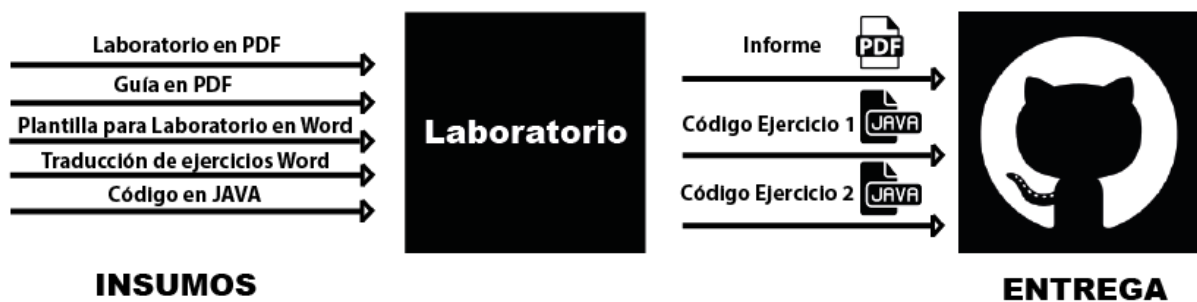
GitHub

Crear un repositorio en su cuenta de GitHub con el nombre `st0245-eafit-suCodigoAqui`. **2.** Crear una carpeta dentro de ese repositorio con el nombre `laboratorios`. **3.** Dentro de la carpeta `laboratorios`, crear una carpeta con nombre `lab02`. **4.** Dentro de la carpeta `lab02`, crear tres carpetas: `informe`, `codigo` y `ejercicioEnLinea`. **5.** Subir el informe pdf a la carpeta `informe`, el código del ejercicio 1 a la carpeta `codigo` y el código del ejercicio en línea a la carpeta `ejercicioEnLinea`. Así:

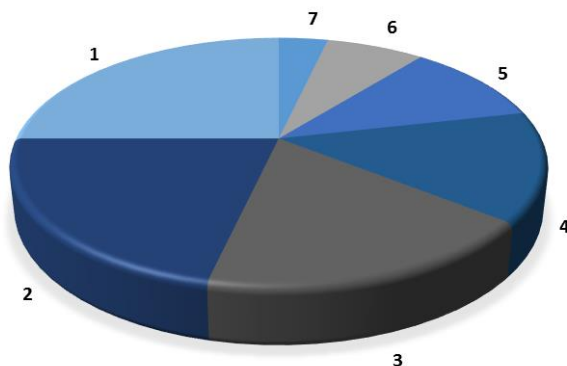
```
st0245-eafit-suCodigoAqui
  laboratorios
    lab01
      informe
      codigo
      ejercicioEnLinea
    lab02
      ...
```

Intercambio de archivos

Los archivos que **ustedes deben entregar** al docente son: **un archivo PDF** con el informe de laboratorio usando la plantilla definida, y **dos códigos**, uno con la solución al numeral 1 y otro al numeral 2 del presente. Todo lo anterior se entrega en **GitHub**.



Porcentajes y criterios de evaluación para el laboratorio



- 1. Simulacro sustentación proyecto
- 2. Análisis de complejidad
- 3. Código de laboratorio
- 4. Simulacro de parcial
- 5. Ejercicio con juez en línea
- 6. Formato PDF y uso GIT
- 7. Documentación código laboratorio

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Ejercicios a resolver

1. Códigos para entregar en GitHub:



En la vida real, la documentación de software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126



PISTA: Lean las diapositivas tituladas “**Data Structures I: O Notation**”, donde encontrarán algoritmos, y en **Eafit Interactiva**, las implementaciones en Java de cada uno



Véase Guía **en Sección 3, numeral 3.4**



Código de laboratorio en **GitHub**. Véase Guía en **Sección 4, numeral 4.24**



Documentación en **HTML**









No se reciben archivos en **.RAR** ni en **.ZIP**

1.1 Extiendan el código existente para medir los tiempos de **Array sum**, **Array Maximum**, **Insertion Sort** y **Merge sort** para arreglos generados aleatoriamente con diferentes tamaños.



NOTA: Todos los ejercicios del numeral 1 deben ser documentados en formato HTML. Véase Guía en **Sección 4, numeral 4.1** “Cómo escribir la documentación HTML de un código usando JavaDoc”

2) Ejercicios en línea sin documentación HTML en GitHub

	Véase Guía en Sección 3, numeral 3.3		No entregar documentación HTML
	Entregar un archivo en .JAVA		No se reciben archivos en .PDF
	Resolver los problemas de CodingBat usando Recursión		Código del ejercicio en línea en GitHub . Véase Guía en Sección 4, numeral 4.24



NOTA: Recuerden que, *si toman la respuesta de alguna fuente*, deben **referenciar** según el **tipo de cita** correspondiente. Véase Guía en **Sección 4, numerales 4.16 y 4.17**

2.1 Resuelvan mínimo 5 ejercicios del nivel **Array 2** de la página **CodingBat**:
<http://codingbat.com/java/Array-2>



NOTA: La complejidad máxima para ejercicios de **Array2** es $O(n)$

2.2 Resuelvan mínimo 5 ejercicios del nivel **Array 3** de la página **CodingBat**:
<http://codingbat.com/java/Array-3>



NOTA: La complejidad máxima para ejercicios de **Array 3** es $O(n^2)$

3) Simulacro de preguntas de sustentación de Proyectos



Véase Guía en **Sección 3,**
Numeral 3.5



Entregar informe de
laboratorio en **PDF**



Usen la **plantilla** para
responder laboratorios



**No apliquen Normas
Icontec para esto**

3.1 En la vida real, para analizar la eficiencia de los algoritmos de forma experimental, debemos probar con problemas de diferente tamaño. El tamaño del problema lo denominamos N . Como un ejemplo, para un algoritmo de ordenamiento, el tamaño del problema es el tamaño del arreglo. Como otro ejemplo, para calcular la serie de Fibonacci, N es el número de términos a calcular.

Teniendo en cuenta lo anterior, completen la siguiente tabla con tiempos en milisegundos

	$N = 100.000$	$N = 1'000.000$	$N = 10'000.000$	$N = 100'000.000$
Array sum				
Array maximum				
Insertion sort				
Merge sort				

3.2 Grafiquen los tiempos que tomó en ejecutarse *array sum*, *array maximum*, *insertion sort* y *merge sort*, para entradas de diferentes tamaños. Si se demora más de un minuto la ejecución, cancela la ejecución y escriba en la tabla “más de 5 minutos”. Grafiquen Tamaño de la Entrada (N) Vs. Tiempo de Ejecución. Utilicen una escala logarítmica para poder graficar correctamente.



PISTA: Véase **Guía sección 4, numeral 4.6** “Cómo usar la escala logarítmica en Microsoft Excel”.



PISTA 2: Si todos los tiempos de un algoritmo dan más de 5 minutos, realice otra tabla, para ese algoritmo, tomando tiempos para arreglos de tamaño 1000, 10000 y 100000.



PISTA 3: Véase **Guía sección 4, numeral 4.4** “Cómo aumentar el tamaño del heap y del stack en Java”



PISTA 4: Véase **Guía sección 4, numeral 4.5** “Cómo visualizar el montículo (heap) y el stack, y el consumo total de memoria de Java”

3.3 ¿Qué concluyen con respecto a los tiempos obtenidos en el laboratorio y los resultados teóricos obtenidos con la notación O?



En la vida real, bases de datos relacionales como *MySQL* guardan los datos ordenados, usando como criterio la llave primaria. Por eso es importante entender la eficiencia de los algoritmos de ordenamiento

3.4 Teniendo en cuenta lo anterior, ¿Qué sucede con *Insertion Sort* para valores grandes de N?



En la vida real, la suma de las ventas de una empresa se hace usando *ArraySum*

3.5 Teniendo en cuenta lo anterior, ¿Qué sucede con *ArraySum* para valores grandes de *N*? ¿Por qué los tiempos no crecen tan rápido como *Insertion Sort*?



En la vida real, el lenguaje de programación *Python* utilizar *Insertion sort* para ordenar arreglos de menos de 100 elementos y *Merge sort* para ordenar arreglos de más de 100 elementos.

3.6 Teniendo en cuenta lo anterior, ¿Qué tan eficiente es *Merge sort* con respecto a *Insertion sort* para arreglos grandes? ¿Qué tan eficiente es *Merge sort* con respecto a *Insertion sort* para arreglos pequeños?

3.7 Expliquen con sus propias palabras cómo funciona el ejercicio *maxSpan* y ¿por qué?



NOTA: Recuerden que debe explicar su implementación en el informe PDF

3.8 Calculen la complejidad de los ejercicios en línea, numerales 2.1 y 2.2, y agréguela al informe PDF



PISTA: Véase *Guía en Sección 4, numeral 4.11* “Cómo escribir la complejidad de un ejercicio en línea”



PISTA 2: Véase *Guía en Sección 4, numeral 4.19* “Ejemplos para calcular la complejidad de un ejercicio de CodingBat”

3.9 Expliquen con sus palabras las variables (qué es ‘*n*’, qué es ‘*m*’, etc.) del cálculo de complejidad del numeral anterior

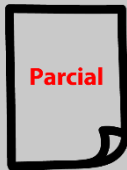
4) Simulacro de Parcial en el informe PDF



PISTA: Véase *Guía en Sección 4, Numeral 4.18* “Respuestas del Quiz”



PISTA 2: Lean las diapositivas tituladas “*Data Structures I: O notation*” y encontrarán la mayoría de las respuestas”



Para este simulacro, agreguen ***sus respuestas*** en el informe PDF.



El día del Parcial no tendrán computador, JAVA o acceso a internet.

1. Supongamos que $P(n,m)$ es una función cuya complejidad asintótica es $O(n \times m)$ y $H(n,m)$ es otra función cuya complejidad asintótica es $O(m \cdot A(n,m))$, donde $A(n,m)$ es otra función. ¿Cuál de las siguientes funciones, definitivamente, NO podría ser la complejidad asintótica de $A(n,m)$ si tenemos en cuenta que $P(n,m) > H(n,m)$ para todo n y para todo m ?

- a) $O(\log n)$
- b) $O(\sqrt{n})$
- c) $O(n+m)$
- d) $O(1)$

2. Dayla sabe que la complejidad asintótica de la función $P(n)$ es: _____. Ayúdala a Dayla a sacar la complejidad asintótica para la función $\text{mystery}(n,m)$.

```
void mystery(int n, int m)
    for(int i = 0; i < m; ++i)
        boolean can = P(n);
        if(can)
```

```
for(int i = 1; i * i <= n; i++)  
    //Hacer algo en O(1)  
else  
    for(int i = 1; i <= n; i++)  
        //Hacer algo en O(1)
```

La complejidad de `mystery(n,m)` es:

- a) $O(m+n)$
- b) $O(m \times n \times \sqrt{n})$
- c) $O(m+n+\sqrt{n})$
- d) $O(m \times n)$

3. El siguiente algoritmo imprime todos los valores de una matriz. El tamaño de la matriz está definida por los parámetros largo y ancho. Teniendo en cuenta que el largo puede ser como máximo 30, mientras que el ancho puede tomar cualquier valor, ¿cuál es su complejidad asintótica en el peor de los casos del algoritmo?

```
01 public void matrices(int[][] m,  
    int largo, int ancho)  
02     for(int i=0; i < largo; i++)  
03         for(int j=0; j < ancho; j++)  
04             print (m[i][j]);
```

- a) $O(\text{largo})$
- b) $O(\text{ancho})$
- c) $O(\text{largo} + \text{ancho})$
- d) $O(\text{ancho} \times \text{ancho})$
- e) $O(1)$

4. Sabemos que $P(n)$ ejecuta pasos y que $D(n)$ ejecuta $n+7$ pasos. ¿Cuál es la complejidad asintótica, en el peor de los casos, de la función $B(n)$?

```
public int B(int n)
    int getP = P(n);
    int ni = 0;
    for(int i = 0; i < n; ++i)
        if(D(i) > 100){
            ni++;
        }
    int nj = getP + D(n) * ni;
    return nj;
```

- a) $O(n^4)$
- b) $O(n^3)$
- c) $O(n^2)$
- d) $O(2^n)$

5. El siguiente algoritmo calcula las tablas de multiplicar del 1 a n. ¿Cuál es su complejidad asintótica en el peor de los casos?

```
public void tablas(int n)
    for(int i=0; i < n; i++)
        for(int j=0; j < n; j++)
            print (i+"*"+j+"="+i*j);
```

- a) $O(n)$
- b) $O(2^i)$
- c) $O(i)$
- d) $O(n^2)$

5. [Ejercicio Opcional] Lectura recomendada



"Quienes se preparan para el ejercicio de una profesión requieren la adquisición de competencias que necesariamente se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades..."

Tomado de <http://bit.ly/2gJKzJD>



Véase Guía en **Sección 3, numeral 3.6 y 4.20** de la Guía Metodológica, "Lectura recomendada" y "Ejemplo para realización de actividades de las Lecturas Recomendadas", respectivamente

Posterior a la lectura del texto "**R.C.T Lee et al., Introducción al análisis y diseño de algoritmos, Capítulo 2. 2005**" realicen las siguientes actividades que les permitirán sumar puntos adicionales:

- a) Escriban un resumen de la lectura que tenga una longitud de 100 a 150 palabras



PISTA: En el siguiente enlace, unos consejos de cómo hacer un buen resumen <http://bit.ly/2knU3Pv>



PISTA 2: [Aquí](#) le explican cómo contar el número de palabras en Microsoft Word

- b) Hagan un mapa conceptual que destaque los principales elementos teóricos.



PISTA: Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en <https://cacao.com/> o <https://www.mindmup.com/#m:new-a-1437527273469>

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co



NOTA: Si desean una lectura adicional en idioma español, consideren la siguiente: “*John Hopcroft et al., Fundamentos de Algoritmia. Capítulo 3: Notación Asintótica. Páginas 11– 98. 1983*” que pueden encontrarla en biblioteca.



NOTA 2: Estas respuestas también deben incluirlas en el informe PDF

6. [Ejercicio Opcional] Trabajo en Equipo y Progreso Gradual



El trabajo en equipo es una exigencia actual del mercado. "Mientras algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques y trabajes bien con otras personas"

Tomado de <http://bit.ly/1B6hUDp>



Véase Guía en **Sección 3, numeral 3.7** y **Sección 4, numerales 4.21 y 4.22 y 4.23** de la Guía Metodológica

- a) Entreguen copia de todas las actas de reunión usando el tablero Kanban, con fecha, hora e integrantes que participaron



PISTA: Véase **Guía en Sección 4, Numeral 4.21** “Ejemplo de cómo hacer actas de trabajo en equipo usando Tablero Kanban”

- b) Entreguen el reporte de *git*, *svn* o *mercurial* con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron



PISTA: Véase Guía en Sección 4, Numeral 4.23 “Cómo generar el historial de cambios en el código de un repositorio que está en svn”

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

- c) Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares



PISTA: Véase Guía en Sección 4, Numeral 4.22 “**Cómo ver el historial de revisión de un archivo en Google Docs**”



NOTA: Estas respuestas también deben incluirlas en el informe PDF

Resumen de ejercicios a resolver

1.1. Extiendan el código existente para medir los tiempos de *Array sum*, *Array Maximum*, *Insertion Sort* y *Merge sort* para arreglos generados aleatoriamente con diferentes tamaños.

2.1 Resuelvan mínimo 5 ejercicios del nivel *Array 2* de la página *CodingBat*:
<http://codingbat.com/java/Array-2>

2.2 Resuelvan mínimo 5 ejercicios del nivel *Array 3* de la página *CodingBat*:
<http://codingbat.com/java/Array-3>

3.1. Completen la siguiente tabla con tiempos en milisegundos, teniendo en cuenta que el tamaño del problema lo denominamos N.

	N = 100.000	N = 1'000.000	N = 10'000.000	N = 100'000.000
Array sum				
Array maximum				
Insertion sort				
Merge sort				

3.2 Grafiquen los tiempos que tomó en ejecutarse *array sum*, *array maximum*, *insertion sort* y *merge sort*, para entradas de diferentes tamaños. Si se demora más de un minuto la ejecución, cancela la ejecución y escriba en la tabla “más de 5 minutos”. Grafiquen Tamaño de la Entrada (N) Vs. Tiempo de Ejecución. Utilicen una escala logarítmica para poder graficar correctamente.

3.3 ¿Qué concluyen con respecto a los tiempos obtenidos en el laboratorio y los resultados teóricos obtenidos con la notación O?

3.4 ¿Qué sucede con *Insertion Sort* para valores grandes de N?

3.5 ¿Qué sucede con *ArraySum* para valores grandes de N? ¿Por qué los tiempos no crecen tan rápido como *Insertion Sort*?

3.6 ¿Qué tan eficiente es *Merge sort* con respecto a *Insertion sort* para arreglos grandes? ¿Qué tan eficiente es *Merge sort* con respecto a *Insertion sort* para arreglos pequeños?

3.7 Expliquen con sus propias palabras cómo funciona el ejercicio *maxSpan* y ¿por qué?

3.8 Calculen la complejidad de los ejercicios en línea, numerales 2.1 y 2.2, y agréguela al informe PDF

3.9 Expliquen con sus palabras las variables (qué es ‘n’, qué es ‘m’, etc.) del cálculo de complejidad del numeral 3.8

4. Simulacro de Parcial

5. Lectura recomendada **[Ejercicio Opcional]**

6. Trabajo en Equipo y Progreso Gradual **[Ejercicio Opcional]**